SOFTMAIL®

พิมพ์ภาพกราฟิกจากจอภาพ

บางครั้งเมื่อเราสร้างภาพกราฟิกสวย ๆ บนจอภาพด้วยเทอร์โบปาสคาลแล้ว ก็อยากจะพิมพ์เก็บไว้ชื่นชมนาน ๆ

นิพันธ์ วงศ์ทองศิริ*

ก่อนที่ผมจะเสนอโปรแกรมย่อยสำหรับการพิมพ์ภาพกราฟิกจากจอภาพ ผมอยากจะให้มาดูรูปแบบการเก็บภาพในแง่ของค่าตัวเลขที่เก็บในหน่วยความจำ และสิ่งที่ผมจะกล่าวถึงต่อไป ผมต้องขอกล่าวก่อนว่าการพิมพ์ภาพกราฟิกและการกล่าวถึงรูปแบบการเก็บค่าตัวเลขในหน่วยความจำ จะเป็นในโหมดของกราฟิกที่มีความละเอียดขนาดกลางของเครื่อง IBM PC/XT/AT (Medium Resolution Mode) ซึ่งมีจุดรายละเอียดขนาด 320 จุด ตามแนวนอนและ 200 จุดตามแนวตั้ง แต่อย่างไรก็ตามการนำโปรแกรมไปประยุกต์ใช้งานในโหมดของกราฟิกที่มีความละเอียดสูง (High Resolution Mode : 640×200 จุด) ก็เพียงแต่แก้ไขโปรแกรมเพียงเล็กน้อยเท่านั้น

IBM ได้กำหนดตำแหน่งของหน่วยความจำเริ่มต้นสำหรับแสดงผลบนจอภาพในกรณีของ Color Graphics Adapter card (CGA) ที่ตำแหน่ง 0B8000 (hex) ไม่ว่าจะอยู่ในโหมดการแสดงตัวอักษร (Text Mode) หรืออยู่ในโหมดแสดงจุดเพื่อประกอบเป็นภาพกราฟิก (Graphics Mode) เพียงแต่ว่ารูปแบบของค่าที่เก็บในหน่วยความจำจะมีความแตกต่างกันในแต่ละโหมดเท่านั้น

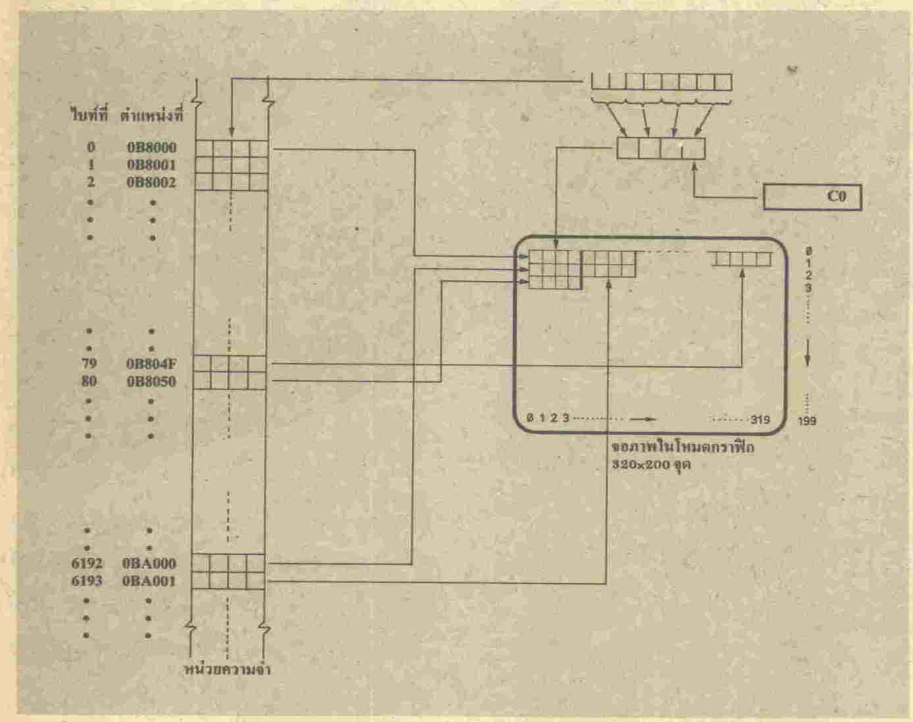
ในโหมดกราฟิกไม่ว่าจะเป็นในแบบ 320 คูณ 200 จุด หรือในแบบ 640 คูณ 200 จุด ต่างก็มีการเก็บค่าในหน่วยความจำโดยแบ่งออกเป็น 2 ส่วน หน่วยความจำส่วนแรกจะเก็บค่าของแต่ละจุดเฉพาะ **แนวแถวคู่** โดยมีตำแหน่งเริ่มต้นที่ **0B8000h** หน่วยความจำส่วนที่สองจะเก็บค่าของแต่ละจุดเฉพาะ **แนวแถวคี่** โดยมีตำแหน่งเริ่มต้นที่ **0BA000h** กล่าวคือ หน่วยความจำส่วนแรกจะเก็บค่าที่แทนแต่ละจุดในแนวแถวที่ 0, 2, 4, 6.... จนถึงแนวแถวที่ 198 และในหน่วยความจำส่วนที่สองจะเก็บค่าที่แทนแต่ละจุดในแนวแถวที่ 1, 3, 5, 7.... จนถึงแนวแถวที่ 199

ในการแทนค่าแต่ละจุดในหน่วยความจำจะใช้หน่วยความจำขนาด 2 บิทต่อ 1 จุด นั่นคือ ในหน่วยความจำขนาด 1 ไบท์ก็จะสามารถเก็บค่าสำหรับแสดงจุดบนจอภาพได้ 4 จุด (แต่ในโหมดของ 640 คูณ 200 จะใช้หน่วยความจำขนาด 1 บิทต่อ 1 จุด) ค่าที่เป็นไปได้ของหน่วยความจำขนาด 2 บิท จะสามารถแสดงค่าแตกต่างกันได้ 4 ค่า คือ ค่า 00, 01, 10 และ 11 (เลขฐานสอง) ซึ่งก็คือในแต่ละจุดจะสามารถแสดงสีได้แตกต่างกัน 4 สี เมื่อมาพิจารณาแถวแนวขวางของจอภาพที่ได้กำหนดไว้จำนวน 320 จุดต่อ 1 แถว (คือจุดที่ 0 ถึงจุดที่ 319) ก็จะต้องใช้หน่วยความจำทั้งสิ้น 80 ไบท์ต่อ 1 แถว (320/4 = 80 ไบท์)

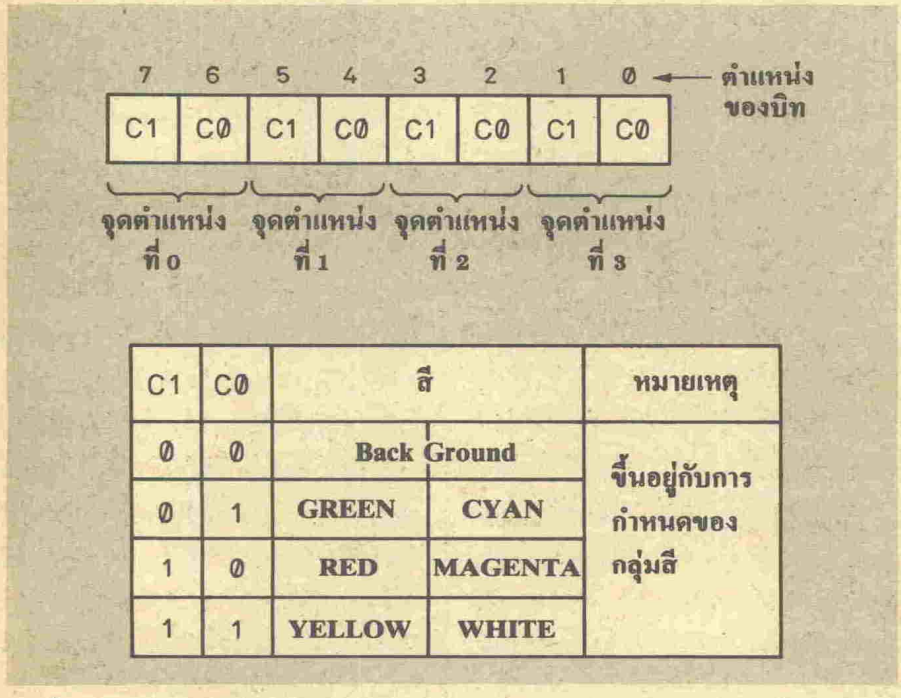
*กองวิเคราะห์และประเมินผลการปฏิบัติการ ฝ่ายแผนปฏิบัติการ ด้านการตลาด การปิโตรเลียมแห่งประเทศไทย

รูปที่ 1 เป็นการแสดงให้เห็นถึงรูปแบบและตำแหน่งของหน่วยความจำที่เก็บ และรูปที่ 2 เป็นตารางความเป็นไปได้ของสีใน 1 จุด และแสดงถึงแต่ละจุดที่ใช้ตำแหน่งของบิทใดในหน่วยความจำ 1 ไบท์

เมื่อพิจารณาจากรูปที่ 2 ในกรณีที่ค่า C1 และ C0 เป็นค่าศูนย์ทั้งคู่ ซึ่งจะเป็นสี Back Ground กล่าวคือ จะเป็นสีเดียวกับสีพื้นที่ไม่สามารถมองเห็นความแตกต่างของสีบนจุดนั่นเอง ในขณะที่ความเป็นไปได้อีก 3 สี จะเป็นสีที่สามารถเห็นได้ชัดจากสีพื้น ฉะนั้นผมจะถือว่าในการพิมพ์ภาพกราฟิกบนจอภาพไปออกเครื่องพิมพ์ ถ้าตำแหน่งของจุดนั้น ๆ มีค่าในหน่วยความจำ 2 บิท ที่มีค่าเป็นศูนย์ทั้งคู่ก็จะไม่มีการพิมพ์จุดดังกล่าว แต่ถ้าหากมีค่าของ 2 บิท ที่แทนค่าจุดนั้นๆ ไม่ว่าจะเป็นค่า 01, 10 หรือ 11 (เลขฐาน 2) ก็จะทำการพิมพ์จุดดังกล่าว



รูปที่ 1 ลักษณะการเก็บค่าบนจอภาพของหน่วยความจำในแบบ Medium



C1	C0	สี		หมายเหตุ
0	0	Back Ground		ขึ้นอยู่กับการกำหนดของกลุ่มสี
0	1	GREEN	CYAN	
1	0	RED	MAGENTA	
1	1	YELLOW	WHITE	

รูปที่ 2 ตารางแสดงค่าของสีในจุดภาพ

หน้าที่ของโปรแกรมส่วนที่ 1

สิ่งที่โปรแกรมจะต้องเกี่ยวข้องส่วนหนึ่งคือการอ่านค่าในหน่วยความจำที่แทนค่าจุด ซึ่งจะเก็บอยู่ในหน่วยความจำที่ตำแหน่งใด ๆ ก็ตาม โดยเริ่มตั้งแต่จุดบนมุมซ้ายบนสุดของจอภาพ (แถวที่ 0 จุดตำแหน่ง 0) โดยอ่านค่าทุกจุดไปจนกระทั่งสิ้นสุด ณ จุดมุมขวาล่างสุดของจอภาพ (แถวที่ 199 จุดตำแหน่งที่ 319) จะเห็นได้ว่ามีข้อยุ่งยากพอสมควรในการคำนวณตำแหน่งของหน่วยความจำที่เก็บค่าแต่ละจุด อย่างที่ผมกล่าวไว้ข้างต้นการเก็บค่าของจุดในหน่วยความจำแต่ละแถวมิได้เก็บค่าติดต่อกันไป ซึ่งมีการแยกส่วนของหน่วยความจำออกเป็นแถวคู่และแถวคี่

ภายใน BIOS (Basic Input/Output System) ของ IBM ได้มีโปรแกรมย่อยสำหรับจัดการในงานส่วนนี้ ทำให้โปรแกรมของเราลดข้อยุ่งยาก การเรียกใช้โปรแกรมย่อยในส่วนนี้คือ การเรียกใช้บริการอินเตอร์รัพต์ เลขที่ 10h หรือเลขที่ 16 ในเลขฐาน 10 ในส่วนของ Function เลขที่ 0Dh หรือเลขที่ 13 ในเลขฐาน 10 การผ่านค่าที่จำเป็นแก่รีจิสเตอร์ที่กำหนดไว้สำหรับอินเตอร์รัพต์ เลขที่นี้สำหรับการอ่านค่าในแต่ละจุด จะแสดงไว้ในรูปที่ 3 ซึ่งค่าที่ได้จะเก็บไว้ที่รีจิสเตอร์ AL เมื่อ

เสร็จสิ้นขบวนการให้บริการโปรแกรมย่อยจากการอินเตอร์รัพต์หมายเลขนี้ ค่าที่ได้จะมีความเป็นไปได้อยู่ 4 ค่า กล่าวคือ ค่า 0 ถึงค่า 3 (ในเลขฐาน 10) หรือค่า 00, 01, 10 และ 11 (ในเลขฐาน 2)

หน้าที่ของโปรแกรมส่วนที่ 2

สิ่งที่โปรแกรมจะต้องเกี่ยวข้องอีกส่วนหนึ่ง ก็คือ การเอาค่าที่อ่านได้ในหน่วยความจำของจอภาพมาเรียงตัวกันสำหรับทำการพิมพ์ ซึ่งการเรียงตัวกันแต่ละจุดนี้จะต้องขึ้นอยู่กับรูปแบบการจัดพิมพ์ที่ถูกกำหนดตามยี่ห้อของเครื่องพิมพ์ ซึ่งอาจจะมีรูปแบบที่เหมือนกันหรือต่างกันในการพิมพ์ในแบบของ bit image

ผมจะขอยกตัวอย่างของเครื่องพิมพ์สำหรับโปรแกรมในบทความนี้ ซึ่งก็คงจะหนีไม่พ้นยี่ห้อ Epson ซึ่งคิดว่าเป็นเครื่องที่ใช้กันแพร่หลายพอสมควร ส่วนยี่ห้ออื่น ๆ ก็จะใช้ได้เหมือนกัน ถ้าหากใช้ Control Code ที่ใช้สำหรับควบคุมเครื่องพิมพ์เหมือนกัน ซึ่งผมก็คิดว่ามีอยู่หลายยี่ห้อทีเดียวที่ใช้ Control Code ที่เหมือนกันกับ Epson แต่สำหรับเครื่องพิมพ์ยี่ห้ออื่น ๆ ที่ไม่เหมือนกับ Epson ถ้าหากสามารถพิมพ์ในโหมดของ bit image ได้แล้ว ก็สามารถดัดแปลงโปรแกรมนี้ได้เช่นกัน โดยจะต้องดูในรายละเอียดจากคู่มือของเครื่องพิมพ์ยี่ห้อนั้น ๆ

INTERRUPT ที่ 10h (16)

Function 0Dh (13)

Read graphics pixel

พารามิเตอร์ที่ต้องผ่านค่า

AH = 0Dh

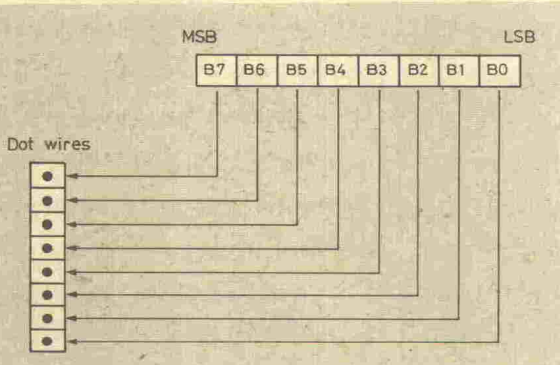
CX = ตำแหน่งของคอลัมน์บนจอภาพ (x-coordinate) ซึ่งมีค่าอยู่ในช่วง 0 ถึง 319 โดยนับจากมุมจอภาพด้านซ้ายไปมุมจอภาพด้านขวา

DX = ตำแหน่งของแถวบนจอภาพ (y-coordinate) ซึ่งมีค่าอยู่ในช่วง 0 ถึง 199 โดยนับจากมุมจอภาพด้านบนลงสู่ด้านล่าง

ค่าที่ได้จากการ Interrupt

AL = ค่าของจุดหรือ pixel value ซึ่งจะมีค่าระหว่าง 0 ถึง 3

รูปที่ 3
ค่าต่าง ๆ ในรีจิสเตอร์เมื่อเรียกอินเตอร์รัพต์ 10 h



รูปที่ 4
ความสัมพันธ์ของหัวเข็มของเครื่องพิมพ์กับตำแหน่งบิทของข้อมูล 1 ไบท์

ข้อกำหนดในโหมด bit image ของเครื่องพิมพ์

เครื่องพิมพ์ Epson ที่ผมใช้ทำการทดสอบโปรแกรมนี้ เป็นรุ่น LQ1500 สำหรับรุ่นอื่น ๆ ผมคิดว่าคงไม่มีปัญหาในการใช้งานโปรแกรมนี้ และ bit image ที่จะทำการพิมพ์จากภาพกราฟิกบนจอภาพนี้ ผมจะใช้ในส่วนของจุดความหนาแน่นการพิมพ์ในแบบธรรมดา ซึ่งในรุ่น LQ1500 จะสามารถพิมพ์ได้ 816 จุดต่อ 1 แถว ซึ่งเหลือเฟือสำหรับ 320 จุดต่อ 1 แถว เมื่อเทียบกับจำนวนจุด 1 แถวของจอภาพ

Control Code ในการสั่งเครื่องพิมพ์ให้อยู่ในโหมดของ bit image ในที่นี้ผมจะขอแสดงรูปแบบที่เขียนในแบบภาษา Pascal (ใช้ Turbo Pascal) คือ

write (lst, chr(27), 'K', chr(n1), chr(n2) ;

write (lst, chr(m), ... ,.....) ;

โดยที่ค่าของ n1 ได้มาจาก

n1 = (number of data) mod 256
= 320 mod 256 (ในที่นี้ค่าของ data คือจำนวนจุด 320 จุดนั่นเอง)
= 64

ค่าของ n2 ได้มาจาก

n2 = INT (Number of data/256)
= INT (320/256)
= 1

สำหรับค่าของ m นั้น คือค่าของไบท์ที่จะส่งตามหลัง ซึ่งในที่นี้มีจำนวนทั้งสิ้น 320 ไบท์ (ที่ผมใช้เครื่องหมายจุดจุดในรูปแบบ ผมหมายถึงมีค่า m ตามมาอีก)

ที่สำคัญคือจะส่งค่า m ด้วยค่าตัวเลขอะไร มาดูความสัมพันธ์ของหัวเข็มกับตำแหน่งแต่ละบิทของข้อมูล 1 ไบท์ ตามรูปที่ 4 ก่อนเถอะครับ ในการพิมพ์โหมด

```
Microsoft (R) Macro Assembler  Version 4.00              9/29/86 09:50:45

Sub-program of Graphics Print Screen                     Page     1-1


                Title Sub-program of Graphics Print Screen
                page ,132
                ;*************************************************************
                ;* Program        : Graphics Print Screen                    *
                ;* Graphics Mode  : Medium resolution mode 320 x 200 dots    *
                ;* written by     : Mr.Niphan Wongthongsiri                  *
                ;* use with       : Turbo Pascal                             *
                ;* date           : September 28,1986                        *
                ;*************************************************************
                ;* example :                                                 *
                ;* ---------                                                 *
                ;* Procedure PrintGraphicScreen(mode:integer); external 'dumpscr.bin'; *
                ;*                                                           *
                ;*          mode = 0  -) normal dot                          *
                ;*          mode = 1  -) reverse dot                         *
                ;*                                                           *
                ;*************************************************************
0000            code    segment word
                assume cs:code
=               mode    equ  [bp+4]
= 001B          esc     equ  27
= 000A          LF      equ  0ah
= 000D          CR      equ  0dh
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0000            scr_dump proc near
0000  55                push bp
0001  8B EC             mov  bp,sp
                         ;-----------------------------------------------------
                         ; save environment
                         ;
0003  50                 push ax
0004  53                 push bx
0005  51                 push cx
0006  52                 push dx
0007  1E                 push ds
0008  EB 04              jmp  short start
                         assume ds:code
000A  00                 row     db  0
000B  0000               column  dw  0
000D  00                 dummy   db  0
000E                start: ;-----------------------------------------------------
                         ; set the 24/180" line spacing of printer
                         ;
000E  B0 1B              mov  al,esc          ; for more information of
0010  E8 00A9 R          call print           ;   this control code
0013  B0 33              mov  al,'3'          ;   please refer to the epson
0015  E8 00A9 R          call print           ;   user guide manual
0018  B0 18              mov  al,24
001A  E8 00A9 R          call print
                         ;-----------------------------------------------------
                         ; initialize the starting row number
                         ;
001D  C6 06 000A R 00    mov row,00
                         ;-----------------------------------------------------
Microsoft (R) Macro Assembler  Version 4.00              9/29/86 09:50:45

Sub-program of Graphics Print Screen                     Page     1-2


                         ; set the printer into the bit image mode.
                         ;
0022  B0 1B              NextRow: mov  al,esc
0024  E8 00A9 R                   call print
0027  B0 4B                       mov  al,'K'
0029  E8 00A9 R                   call print
002C  B0 40                       mov  al,320 mod 256       ; 320 mod 256 = 64
002E  E8 00A9 R                   call print
0031  B0 01                       mov  al,320/256           ; 320/256 = 1
0033  E8 00A9 R                   call print
                                  ;-----------------------------------------------------
                                  ; initialize a column number
                                  ;
0036  C7 06 0008 R 0000           mov  column,0000
                                  ;-----------------------------------------------------
                                  ; set a bit-counter & clear a dummy byte
                                  ;
003C  B9 0008            NextCol: mov  cx,8             ; cx is now contained a bit-counter
003F  C6 06 000D R 00             mov  dummy,00
0044  B6 00                       mov  dh,00
0046  8A 16 000A R                mov  dl,row           ; dx is now contained a Y coordinate
004A  51                 NextDot: push cx               ; save temporarily a bit-counter
004B  8B 0E 0008 R                mov  cx,column        ; cx is now contained a X coordinate
004F  D0 26 000D R                shl  dummy,1
0053  B4 0D                       mov  ah,0dh           ; 0dh is a read graphics pixel function
0055  CD 10                       int  10h              ; interrupt video mode
0057  3C 00                       cmp  al,00            ; al is now contained a pixel value
0059  74 05                       je   short NotDot
0058  80 0E 000D R 01             or   dummy,01h        ; set dot
0060  42                 NotDot:  inc  dx               ; row = row+1
0061  59                         pop  cx               ; restore a bit-counter
0062  E2 E6                       loop NextDot          ; cx= cx-1;
                                                        ; if bit-counter >0 then read
                                                          the next row dot
0064  A0 000D R                   mov  al,dummy         ; else print it
0067  80 7E 04 00                 cmp  byte ptr mode,0  ; Is it a reverse mode ?
006B  74 02                       je   short write      ; if not then goto write
006D  F6 D0                       not  al               ; else reverse it
006F  E8 00A9 R          write:   call print
0072  FF 06 0008 R                inc  column
```

รูปที่ 5 โปรแกรมพิมพ์ภาพกราฟิกจากจอภาพ

bit image ในแบบความหนาแน่นธรรมดา หัวพิมพ์ของเครื่องพิมพ์จะทำการพิมพ์ได้ครั้งละ 8 จุด ตามแนวตั้ง จากภาพจะเห็นได้ว่าเมื่อทำการส่งค่า 1 ไบท์ เครื่องพิมพ์จะตีความหมายโดยเอาค่าบิทที่ 7 แทนหัวเข็มบนสุด และบิทที่ 0 แทนหัวเข็มล่างสุดของหัวพิมพ์ เมื่อหันกลับมาเทียบกับจอภาพ ซึ่งมีจุดตามแนวตั้งทั้งสิ้น 200 จุด ฉะนั้นแต่ละแนวตั้งหรือ 1 คอลัมน์จะต้องพิมพ์ 25 ครั้ง (200/8 = 25) แต่ในการพิมพ์ของเครื่องพิมพ์ตามธรรมดาจะทำการพิมพ์จากซ้ายไปขวาจนสุดและขึ้นบรรทัดใหม่ โปรแกรมที่ทำการอ่านค่าในหน่วยความจำก็เป็นการอ่านค่าแต่ละจุดจากทางซ้ายของจอภาพไปทางขวาของจอภาพเช่นเดียวกัน แต่การอ่านจะอ่านทีละ 8 แถวใน 1 คอลัมน์ กล่าวคือเริ่มอ่านครั้งแรกก็จะอ่านค่าของจุดตามตำแหน่งของจอภาพที่ตำแหน่ง (0,0) (1,0) (2,0)...(7,0) (ตัวเลขหน้าภายในวงเล็บคือ แถวที่และตามคอลัมน์ที่) เมื่อได้ครบก็จะทำการแปลงค่าเป็นค่า 1 ไบท์ส่งไปยังเครื่องพิมพ์ จากนั้นก็จะอ่านตำแหน่งถัดไปทางขวา คือ (0,1) (1,1) (2,1)...(7,1) เรื่อยไปจนกระทั่งถึงขวาสุดของจอภาพ และทำการเริ่มอ่านอีก 8 แถวถัดไป คือแถวที่ 8 ถึงแถวที่ 15 เรื่อยไป ซึ่งจะเป็นจำนวนแถวการอ่านทั้งสิ้น 25 ครั้ง (อ่านทีละ 8 แถว) โดยแต่ละแถวของการอ่านจะมีการอ่านภายในแถวทั้งสิ้น 320 ครั้ง หรือได้ค่าเป็นจำนวน 320 ไบท์ ส่งไปที่เครื่องพิมพ์ใน 1 แถวการพิมพ์

โปรแกรมย่อยที่ถูกเรียกจากโปรแกรมหลัก

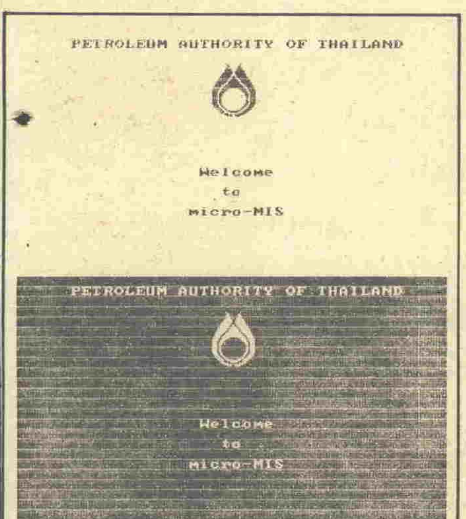
โปรแกรมที่ผมเขียนขึ้นนี้ เขียนด้วยภาษาแอสเซมบลีโดยจะถูกเรียกใช้งานจากภาษาปาสคาล ซึ่งในที่นี้ผมขอใช้เทอร์โบปาสคาล

การใช้งานจะสามารถเรียกใช้ได้ 2 โหมดคือ

เมื่อ mode มีค่า = 0 คือ จะพิมพ์

```
0076  81 3E 0008 R 0140   cmp  column,320       ; Is it a last dot of that line on the screen ?
007C  75 BE               jne  short NextCol    ; if not then read the next column dot
                          ;--------------------------------------------------------
                          ; move a print-head of printer to the new line
                          ;
007E  B0 0D               mov  al,CR            ; send a carriage return
0080  E8 00A9 R           call print
0083  B0 0A               mov  al,LF            ; send a line feed
0085  E8 00A9 R           call print
0088  80 06 000A R 08     add  row,8            ; skip to the next 8 row
008D  80 3E 000A R C8     cmp  row,200          ; Is it the last row of screen ?
0092  72 8E               jb   short NextRow    ; if not then process continue
                          ;--------------------------------------------------------
                          ; restore the 1/6" line spacing of printer
                          ;

Microsoft (R) Macro Assembler  Version 4.00                 9/29/86 09:50:45

Sub-program of Graphics Print Screen                        Page    1-3


0094  B0 1B               mov  al,esc           ; see the epson user guide manual
0096  E8 00A9 R           call print            ;   for the detail of information
0099  B0 32               mov  al,'2'
009B  E8 00A9 R           call print
                          ;--------------------------------------------------------
                          ; restore environment
                          ;
009E  1F                  pop  ds
009F  5A                  pop  dx
00A0  59                  pop  cx
00A1  5B                  pop  bx
00A2  58                  pop  ax
                          ;--------------------------------------------------------
00A3  8B E5               mov  sp,bp
00A5  5D                  pop  bp
00A6  C2 0002             ret  2
                 Scr_dump endp
                          ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::
00A9                print proc near
00A9  52                  push dx
00AA  B4 00               mov  ah,00
00AC  BA 0000             mov  dx,00
00AF  CD 17               int  17h         ; see the IBM technical manual for
00B1  5A                  pop  dx          ; the detail of interrupt No. 17h
00B2  C3                  ret
                    print endp
                          ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::
00B3                 code ends
                          end
```

```
Microsoft (R) Macro Assembler  Version 4.00                 9/29/86 09:50:45

Sub-program of Graphics Print Screen                        Symbols-1

Segments and Groups:

                N a m e             Size        Align       Combine Class

CODE . . . . . . . . . . . . . .    00B3        WORD        NONE

Symbols:

                N a m e             Type        Value       Attr

COLUMN . . . . . . . . . . . . .    L WORD      000B        CODE
CR . . . . . . . . . . . . . . .    Number      000D

DUMMY  . . . . . . . . . . . . .    L BYTE      0000        CODE

ESC  . . . . . . . . . . . . . .    Number      001B

LF . . . . . . . . . . . . . . .    Number      000A

MODE . . . . . . . . . . . . . .    Text  [bp+4]

NEXTCOL  . . . . . . . . . . . .    L NEAR      003C        CODE
NEXTDOT  . . . . . . . . . . . .    L NEAR      004A        CODE
NEXTROW  . . . . . . . . . . . .    L NEAR      0022        CODE
NOTDOT . . . . . . . . . . . . .    L NEAR      0060        CODE

PRINT  . . . . . . . . . . . . .    N PROC      00A9        CODE        Length = 000A

ROW  . . . . . . . . . . . . . .    L BYTE      000A        CODE

SCR_DUMP . . . . . . . . . . . .    N PROC      0000        CODE        Length = 00A9
START  . . . . . . . . . . . . .    L NEAR      000E        CODE

WRITE  . . . . . . . . . . . . .    L NEAR      006F        CODE


   137 Source  Lines
   137 Total   Lines
    37 Symbols

  49348 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```



รูปที่ 6 ภาพตัวอย่างที่ได้จากเครื่องพิมพ์

ภาพกราฟิกแบบที่เห็นบนจอภาพ โดยจุดที่เป็นสีจะแทนจุดในการพิมพ์ จุดที่เป็นสีพื้น (background) ก็จะไม่ทำการพิมพ์

เมื่อ mode มีค่า = 1 คือจะทำการพิมพ์ภาพกราฟิกในลักษณะตรงกันข้ามกับ mode 0

ผมได้เขียนโปรแกรมซึ่งแสดงในส่วนของ object code ไว้ด้วย ในรูปที่ 5 สำหรับรูปที่ 6 เป็นตัวอย่างการใช้งานจาก Turbo Pascal ซึ่งพิมพ์จากเครื่อง Epson สำหรับการพิมพ์ภาพจากกราฟิกโหมดในแบบ 640 คูณ 200 นั้นก็สามารถใช้หลักการเดียวกันได้ เพียงแต่แก้ไขโปรแกรมเล็กน้อยเท่านั้น หรือถ้าหากจะเขียนเพิ่มเพื่อให้โปรแกรมสามารถตรวจสอบดูว่าขณะที่จะทำการพิมพ์กำลังแสดงภาพกราฟิกในโหมดใด และก็ทำการอ่านค่าในรูปแบบของจุดในโหมดนั้น ๆ เพื่อทำการพิมพ์ ก็จะเพิ่มความสามารถของโปรแกรมให้ดีขึ้นอีก ลองคิดดูซิครับ ผมรับรองว่าไม่ยากเลย

หมายเหตุ

เท่าที่ผมตรวจสอบดูโปรแกรมนี้ สามารถใช้ได้กับเครื่องพิมพ์ยี่ห้ออื่น ๆ ที่นอกเหนือจาก Epson ได้ทันทีโดยไม่ต้องแก้ไข คือ NEC รุ่น P5, SUPER-5 รุ่น EP-1201 และคงจะมีอีกหลายยี่ห้อ แต่ผมไม่สามารถกล่าวอ้างได้เพราะไม่มีคู่มืออ้างอิง